Title:	Exploring OpenSNAPI Use Cases and Evolving Requirements
Author(s):	Williams, Brody Kyle Poole, Stephen Wayne Poole, Wendy Kinton
Intended for:	HPC Symposium 2021
Issued:	2021-08-18

Los Alamos
NATIONAL LABORATORY

Exploring OpenSNAPI Use Cases and Evolving Requirements

Brody Williams*^,
Steve Poole ‡, Wendy Poole‡

August 19th, 2021

* Presenter, ^ Texas Tech University, ‡ Los Alamos National Laboratory

LA-UR-21-XXXXX

Exploring OpenSNAPI Use Cases and Evolving Requirements

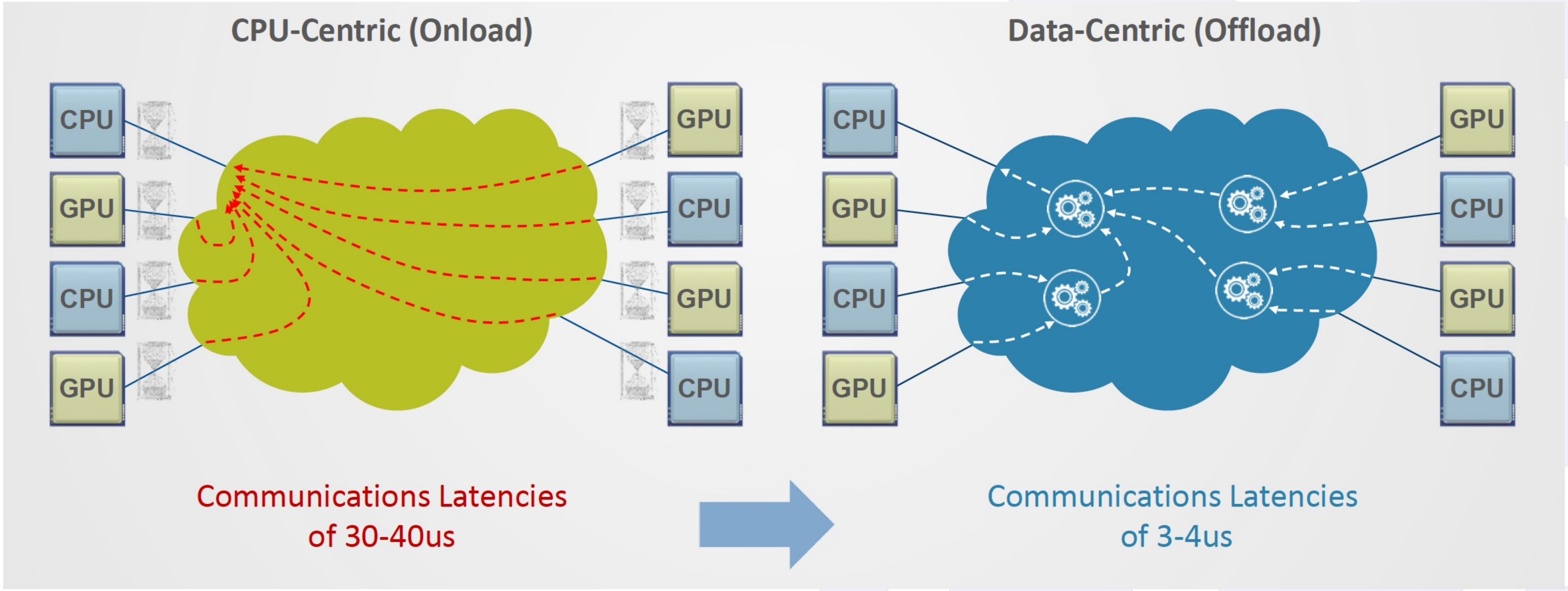
Emerging system architectures are rapidly transforming in order to meet shifting requirements. Motivated by expanding data volumes, energy efficiency concerns, and the omnipresent need to improve performance, architectures are increasingly adopting a data-centric approach. At the core of this concept is the goal of minimizing data motion and instead processing data in-situ to the greatest degree possible.

Therefore, data-centric designs, in contrast to conventional CPU-centric models, typically distribute compute capabilities throughout the architecture. As part of this paradigm shift, a novel class of devices known as data processing units (DPUs), alongside CPUs and GPUs, are quickly forming a third pillar of data-centric systems. These devices, which include smart network adapters and switches, seek to offload computation on data at the network edge as well as in-flight within the network fabric.

The Open Smart Network API (OpenSNAPI) project seeks to develop a unified API for DPU devices. In our previous talks, we introduced the OpenSNAPI project and detailed our investigations regarding the viability of offloading compute intensive kernels to BlueField DPUs. In contrast, in this talk we detail our efforts to offload application-level file I/O to the DPU. We also discuss plans and early efforts to explore in-network compute capabilities. Finally, we describe our observations with respect to the evolving design of OpenSNAPI.

LA-UR-21-27657

OpenSNAPI - Motivation



https://developer.arm.com/-/media/Files/downloads/hpc/files/Arm-HPC-UG-ISC18/GoingArm_SC18_Mellanox.pdf?revision=8681b9d7-8a8c-411d-950d-a0b1a8c7c1f9&hash=982119B5C20963F8FF2B40870D8E3725

OpenSNAPI

- Open Smart Network API
  - SmartNICs/Data Processing Units (DPUs)
  - In-fabric components (switches)

- Example SmartNICs/DPUs
  - NVIDIA (Mellanox) BlueField
  - Broadcom Stingray
  - Marvell Octeon
  - Xilinx Alveo

UCF Unified Communication Framework

https://www.nvidia.com/en-us/networking/products/data-processing-unit/

OpenSNAPI - Previous Work

- Investigated application-level offloading of compute kernels and communication routines to first generation BlueField DPUs

- Benchmarks/Applications
  - PENNANT
  - GUPs
  - Two DoD proxy apps

- Takeaway: Careful analysis / task-level parallelism required

OpenSNAPI Use Case – Offloading File I/O

- BigSort Benchmark[1] (LLNL)

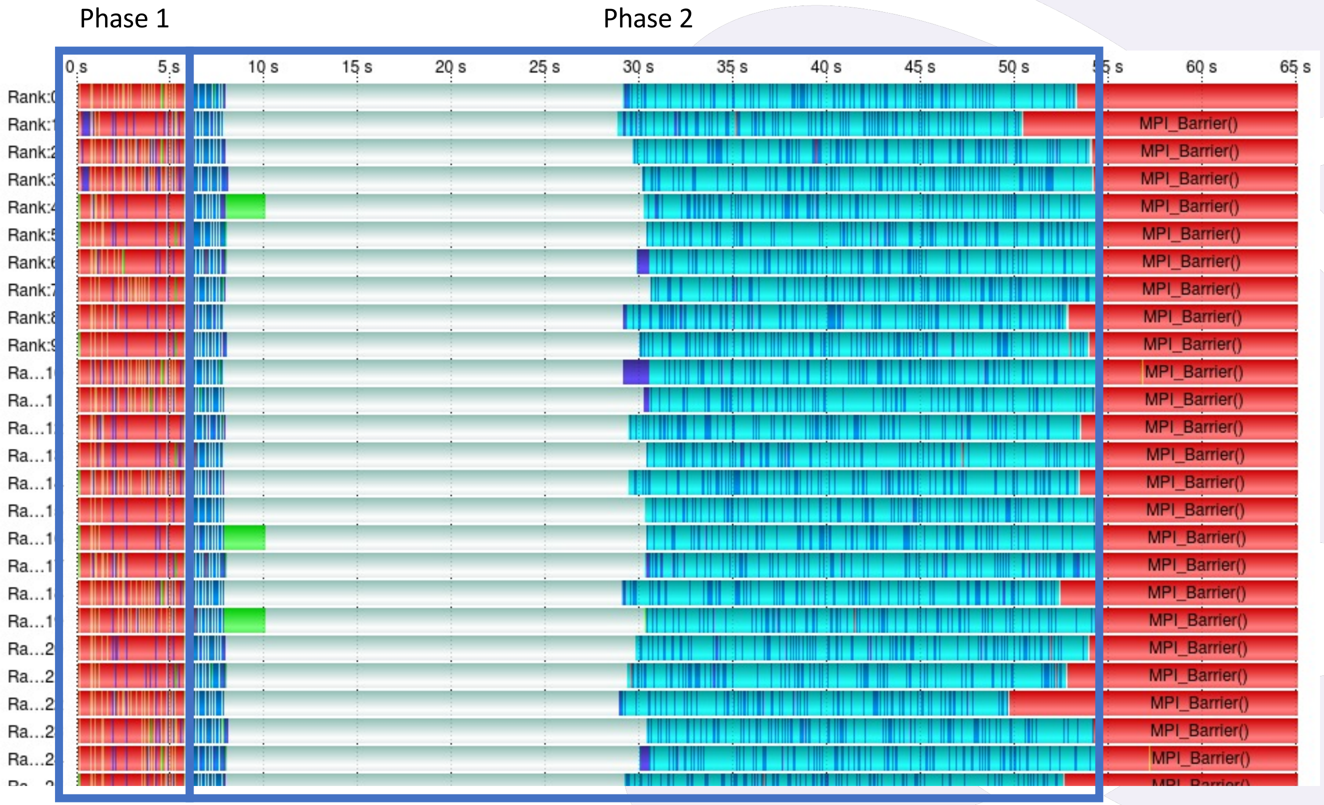
- Parallel sort of 64-bit integer values
  - Total data size may exceed available memory resources
  - Integer operations, all-to-all communication, file I/O
  - MPI + OpenMP

- Two Phases:
  I. Sort values from distinct data segments into num_ranks bins; transfer binned values to appropriate destination via MPI_AlltoAllv()
  II. Perform sort of local data into proper sequence
     - Interleaved computation and file I/O

[1]https://asc.llnl.gov/coral-2-benchmarks

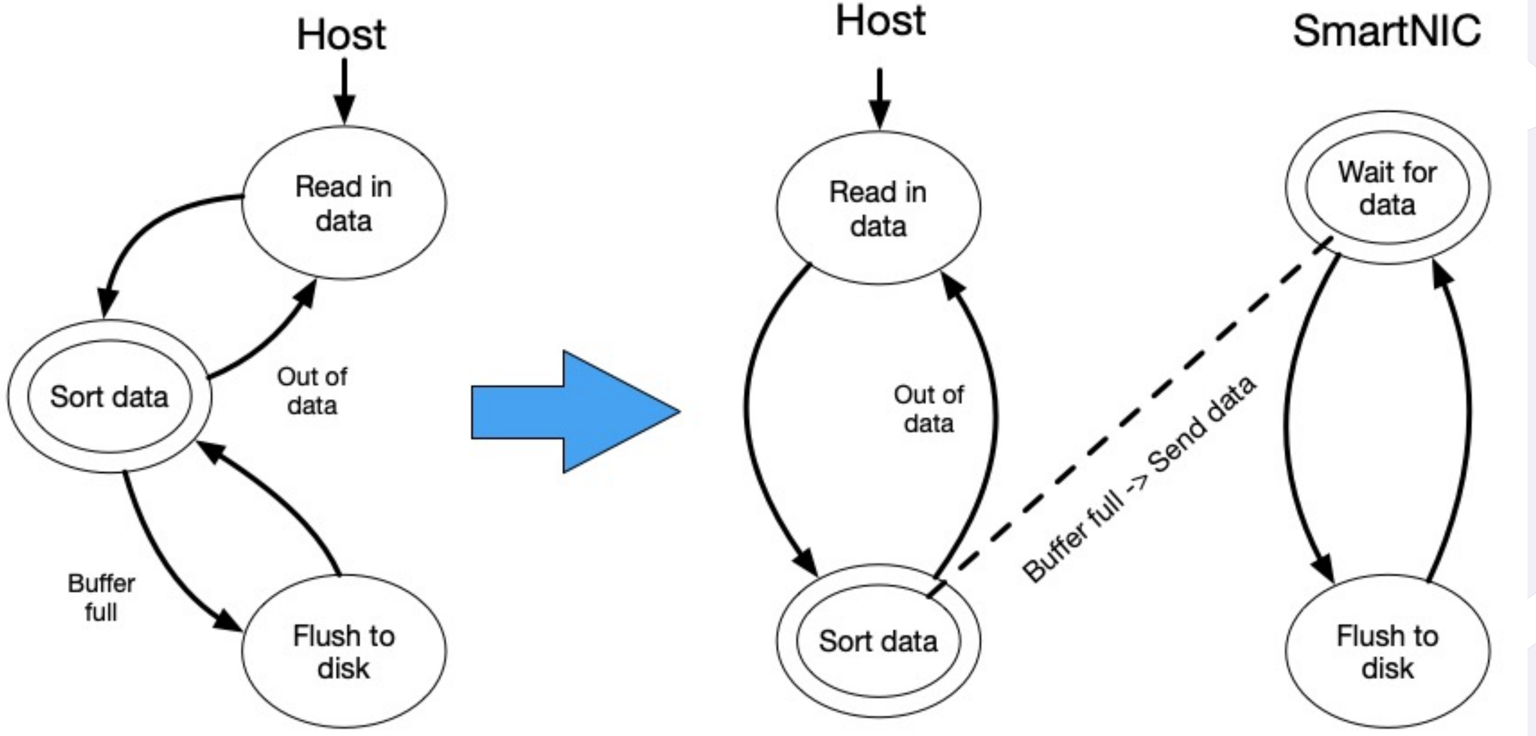
OpenSNAPI Use Case – Offloading File I/O

- **MPI Calls**
- **QuickSort**
- **Data Merge**
- **Write()**



OpenSNAPI Use Case – Offloading File I/O

Phase Two Abstraction



OpenSNAPI Use Case – Offloading File I/O

- No accessible NFS from Ghost/Nymeria

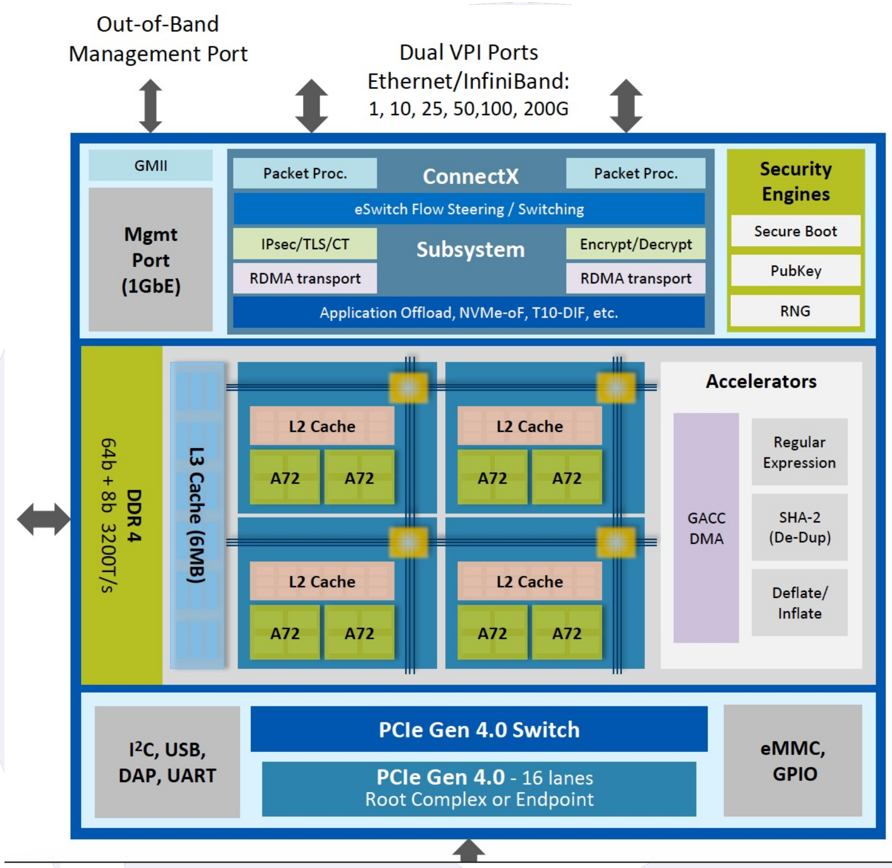
- Proxy evaluation on Capulin
  - Vanilla: 4 host processes, 1 ppn
  - Offload: 4 host + 4 simulated DPU processes, 1 ppn

	Test Case			
	1	**2**	**3**	**4**
Elements (Size in GiB)	8192 (~ 640KiB)	134217728 (1GiB)	536890912 (~4GiB)	536890912 (~4GiB)
Memory Size	2KiB	2MiB	2MiB	20MiB
Page Size	512 B	4 KiB	8 KiB	800 KiB
Original Time (s)	0.475	111.870	396.956	36.829
Offload Time	0.353	106.061	384.814	33.201
Improvement (%)	25.655	5.191	3.058	9.849

Los Alamos NATIONAL LABORATORY

Regular Expression Analysis

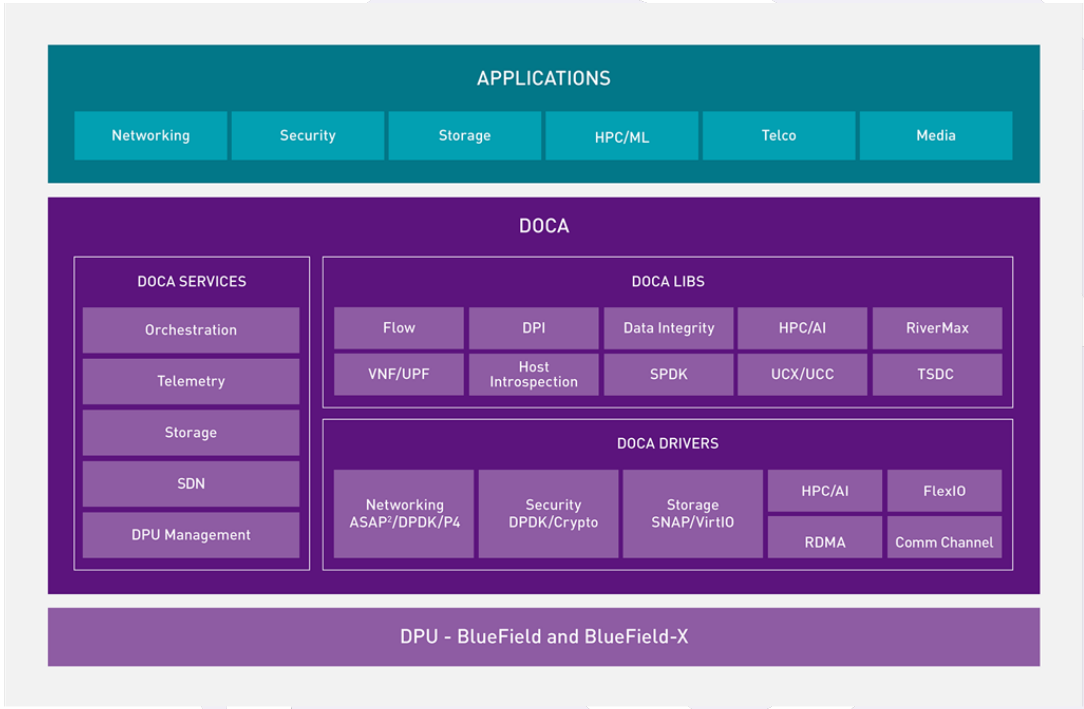
- What useful work can we do on in-flight data?
- Extract information from packets and process using RegEx engine
- Possible Use Cases
  - Security
  - Data analytics
  - …



https://www.hpcadvisorycouncil.com/events/2019/australia-conference/pdf/day-two/A_Ambastha_Mellanox_IB_InNetworkComp_Wed_280819.pdf

Regular Expression Analysis

- Data Center Infrastructure-on-a-Chip Architecture (DOCA)
  - NVIDIA's BlueField SDK
- Deep Packet Inspection (DPI)
- FLOW
- Concerns:
  - InfiniBand support
  - "Hackability"

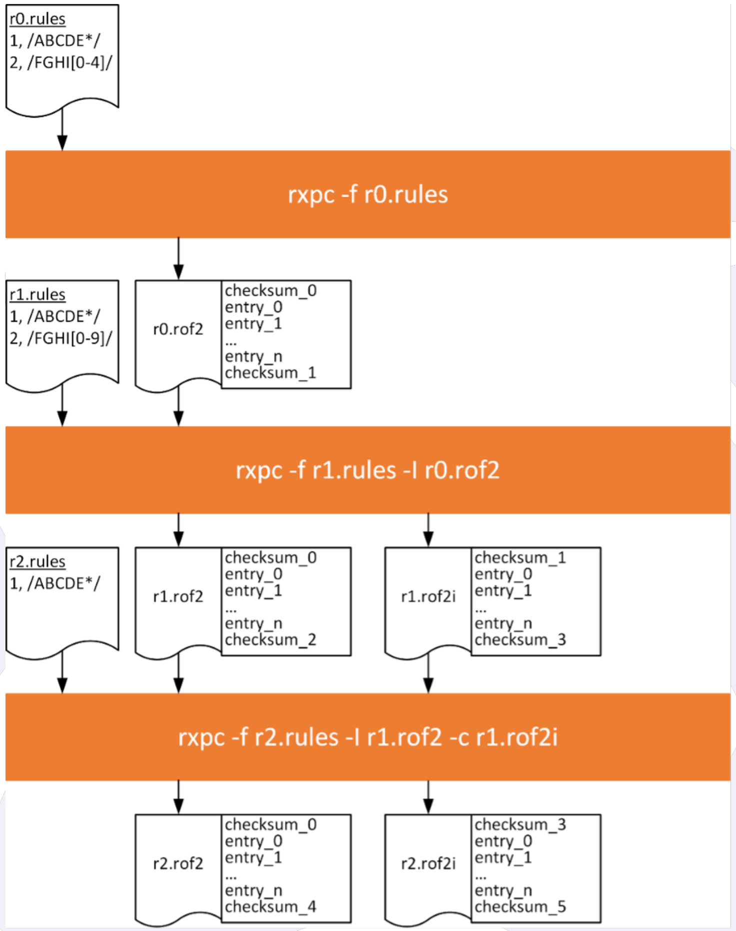


https://developer.nvidia.com/networking/doca

Regular Expression Analysis

- RXP compiler
  - Perl Compatible Regular Expressions (PCRE)
  - Produces .rof object files for configuring RegEx engine
  - Supports dynamic, incremental, updates

- LANL's BlueField-2s are not yet online
  - We are currently building a "BlueField" VM for experimenting with the RXP compiler



https://docs.nvidia.com/doca/sdk/rxp-compiler/index.html

Regular Expression Analysis – Packet Capture

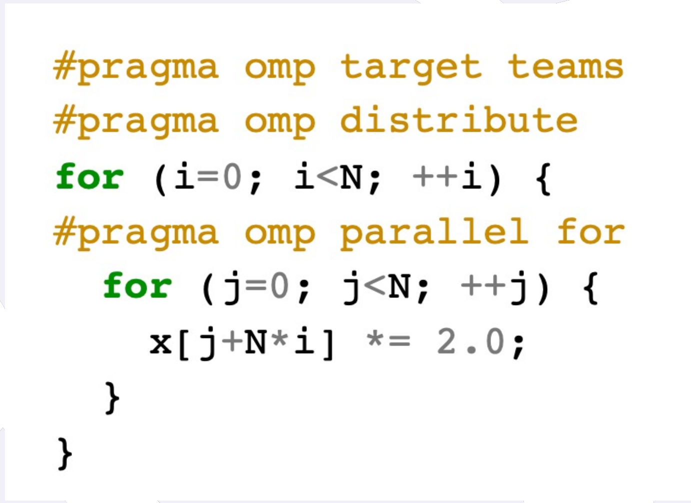
- LibPCAP
  - Seem to be the de-facto standard
    - Utilized by Snort, Suricata, Wireshark
  - Pros:
    - Simple C language API
    - InfiniBand support (IBdump)
  - Cons:
    - Kernel-space utility
    - RegEx packet filtering process
    - Promiscuous mode privileges

- NVIDIA team is currently investigating other options

OpenSNAPI - Design Considerations

- Current OpenSHMEM/MPI MPMD programming model is not sustainable
  - Development process is complex and time-consuming
    - Necessitates significant refactoring for existing codes
  - Resulting applications are difficult to debug

- OpenSNAPI Goals
  - Provide simple, inline, offload API
    - Runtime library, compiler directives, etc.
  - Abstract away complex details
  - Maintain flexibility
    - Developer should be able to offload arbitrary code segments



```
#pragma omp target teams
#pragma omp distribute
for (i=0; i<N; ++i) {
#pragma omp parallel for
  for (j=0; j<N; ++j) {
    x[j+N*i] *= 2.0;
  }
}
```

https://ecpannualmeeting.com/assets/overview/sessions/ff2020%20ECP-Tutorial-with-ECP-template.pdf

OpenSNAPI - Design Considerations

- To offload or not to offload?
  - A mechanism for guiding the what, when, and where is needed for OpenSNAPI
    - Naïve DPU offloading will not end well

- For the above, we need to be able to quantify offload cost/benefit
  - How long is needed to move the necessary data?
  - How long will it take X device to do the compute?
  - Is there significant work to keep the host busy during this time?

- Detailed application profiling is time-consuming
  - May be untenable for large applications

Acknowledgements

- This work made possible by:
  - Steve & Wendy Poole
  - Julie Wiens
  - NVIDIA Team
  - U.S. Department of Defense

- UCF Consortium and associated project links:
  - UCF Consortium: https://www.ucfconsortium.org/
  - OpenSNAPI: https://www.ucfconsortium.org/projects/opensnapi/
  - (Open)HPCA: https://www.ucfconsortium.org/projects/hpca-benchmark/

Thank You!

Questions?